10/11/22

TERMWORK - 3

Problem Definition

Implement the Distance Vector Routing Algorithm.

Objectives

To implement Distance Vector Routing algorithm to find suitable path for transmission.

Theory

- An internet is a combination of networks connected by routers.
- When a packet goes from source to destination, it will probably pass through many routers until it reaches the router attached to the destination network.
- A router consults a routing table when a packet is ready to be forwarded.
- The routing table specifies the optimum path for the packet.
- The table can be either static or dynamic.

Static Table : Does not change frequently.
Dynamic Table : Updated automatically where there is a change somewhere in the internet.

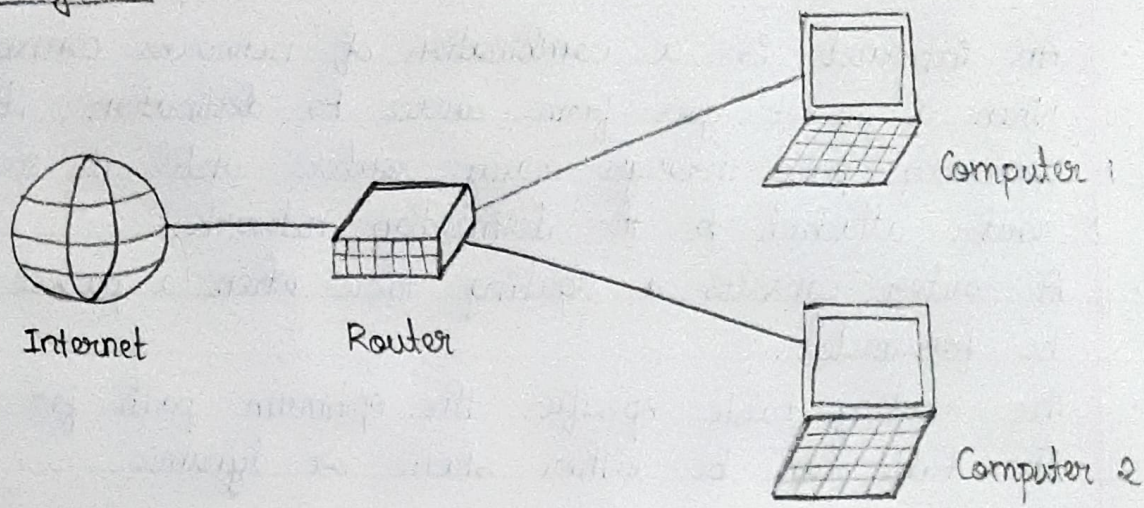
Dynamic Routing Table

- An internet needs dynamic routing table in order to incorporate changes into the table whenever there is a change in internet.
- They need to be updated when the route is down, or when a better route is created.

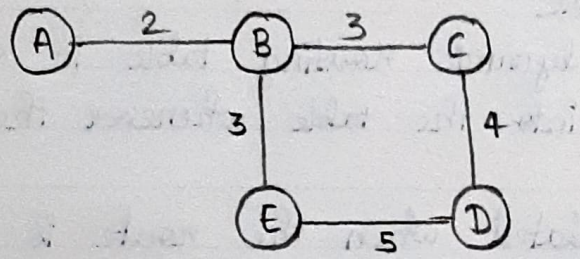
Routing Protocols

- Routing Protocol is a combination of rules & procedures that lets routers in the internet inform one another of changes.
- Have been created in response to the demand for dynamic routing tables.
- It allows router to share whatever they know about the internet or neighbourhood.

Diagram



Computation



A's table

A	0
B	2
C	∞
D	∞
E	∞

B's table

A	2
B	0
C	3
D	∞
E	5

C's table

A	∞
B	3
C	0
D	4
E	∞

Unicast : There is one source & one destination.
Multicast : More than two addresses may be involved.

Metric

- A router receives a packet from a network & passes it to another network.
- A router is usually attached to several networks.
- When it receives a packet, to which network should it pass the packet? This decision is based on optimization.
- A metric is a cost assigned for passing through a network.

Interior Routing : Routing inside an autonomous system.
Exterior Routing : Routing between autonomous systems.

Distance Vector Routing

- Three key points in understanding the working of Distance Vector Routing are as follows :
  i) Sharing knowledge about the entire autonomous system : Each router shares its knowledge about the entire autonomous system with its neighbours.
  ii) Sharing only with neighbours : Each router sends its knowledge only to neighbours.
  iii) Sharing at regular intervals : Each router sends its knowledge to its neighbours at fixed intervals.

- Adaptive / Dynamic algorithm : Route decisions will change dynamically in milliseconds.
- Each router maintains a table called Vector. Vector contains number of hops & delays. Table has the best known

D's table

A	∞
B	∞
C	4
D	0
E	5

E's table

A	∞
B	3
C	∞
D	5
E	0

distance for each router. Tables are updated by exchanging information with neighbours.

- Each router knows the best distance to reach another router.
- Also known as Bellman Ford Algorithm.

Algorithm

1. Send my routing table to all my neighbours whenever my link table changes.
2. When I get a routing table from a neighbour on port P with link metric M:
   * add L to each of the neighbour's metrics.
   * for each entry (D, P', M') in the updated neighbour's table.
     i) if I do not have an entry for D, add (D, P, M') to my routing table
     ii) if I have an entry for D with metric M", add (D, P, M') to my routing table if M' < M"
3. If my routing table has changed, send all the new entries to all my neighbours.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#define nul 1000
#define nodes 10
int no;
struct node
{
    int a[nodes][4];
} router[nodes];

void init(int r)
{
    int i;
    for(i=1; i<=no; i++)
    {
        router[r].a[i][1] = i;
        router[r].a[i][2] = 999;
        router[r].a[i][3] = nul;
    }
    router[r].a[r][2] = 0;
    router[r].a[r][3] = r;
}

void inp(int r)
{
    int i;
    printf("\n Enter dist from the node %d to other nodes", r);
    printf("\n Pls enter 999 if there is no direct route \n");
```

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#define nul 1000
#define nodes 10
int no;
struct node
{
    int a[nodes][4];
} router[nodes];

void init(int r)
{
    int i;
    for(i=1; i<=no; i++)
    {
        router[r].a[i][1] = i;
        router[r].a[i][2] = 999;
        router[r].a[i][3] = nul;
    }
    router[r].a[r][2] = 0;
    router[r].a[r][3] = r;
}

void inp(int r)
{
    int i;
    printf("\n Enter dist from the node %d to other nodes", r);
    printf("\n Pls enter 999 if there is no direct route \n");
```

```
        for(i=1 ; i<=no ; i++)
        {
            if (i != r)
            {
                printf("\n Enter dist to the node %d :" i);
                scanf("%d", &router[r].a[i][2]);
                router[r].a[i][3] = i;
            }
        }
    }

void display (int r)
{
    int i, j;
    printf("\n\n The routing table for node %d is:", r);
    for (i=1; i<=no ; i++)
    {
        if ( router[r].a[i][2] >= 999)
            printf("\n\t\t\t %d \t no link \t no hop",
              router[r].a[i][1]);
        else
            printf("\n\t\t\t %d \t %d \t\t %d, router[r].a[i][1],
              router[r].a[i][2], router[r].a[i][3]);
    }
}

void dv_algo( int r)
{
    int i, j, z;
```

```
    for(i=1; i<=no; i++)
    {
        if(router[x].a[i][2]!=999 && router[x].a[i][2]!=0)
        {
            for(j=1; j<=no; j++)
            {
                z = router[x].a[i][2] + router[i].a[j][2];
                if(router[x].a[j][2] > z)
                {
                    router[x].a[j][2] = z;
                    router[x].a[j][3] = i;
                }
            }
        }
    }
}

int main()
{
    int i, j, x, y;
    char choice = 'y';
    printf("Enter the number of nodes: ");
    scanf("%d", &no);
    for(i=1; i<=no; i++)
    {
        init(i);
        inp(i);
    }
    printf("\n The configuration of the nodes after initialization
     is as follows:");
```

```
for(i=1; i<=no ; i++)
    display(i);
for(i=1; i<=no ; i++)
    dv_algo(i);
printf("\n The configuration of the nodes after computation
 is as follows :");
for(i=1; i<=no ; i++)
    display(i);

while( choice != 'n')
{
    printf("\n Enter the nodes btn which shortest path
     is found : \n");
    scanf("%d %d",&x, &y);
    printf("\n The length of the shortest path is %d",
     router[x].a[y][2]);
    printf("\n\n Continue ? (y/n) : ");
    scanf("%s", &choice);
}
}
```

Conclusion

We successfully implemented the distance vector routing algorithm.

Learning Outcomes

- Understand the concept of Distance Vector Routing.
- Implement 'C' program to find the shortest path between the two nodes.

References.

Behrouz Forouzan - Data Communications & Networking, McGraw Hill Edition.